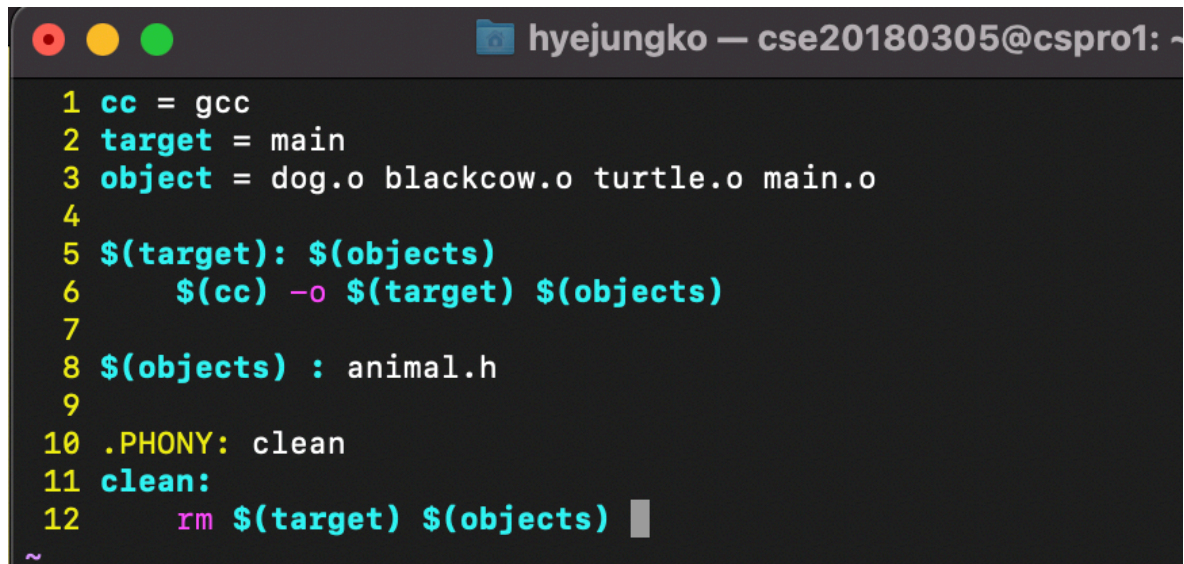


1. 실습 결과 화면을 첨부하시오.

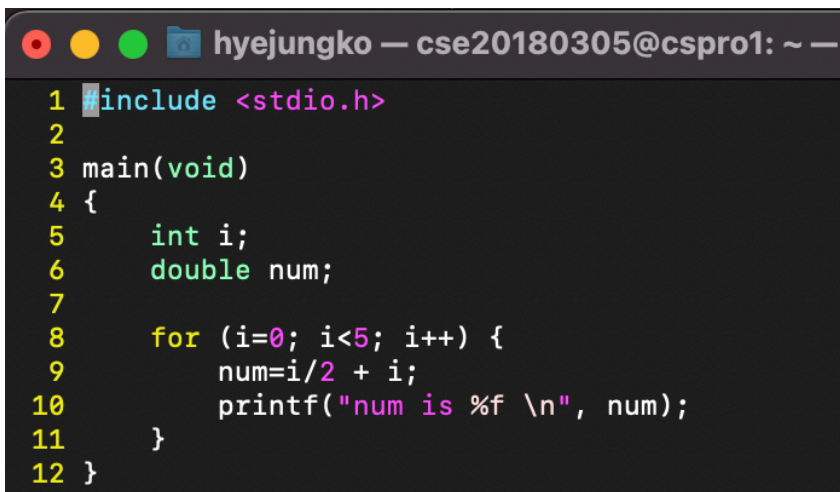
1) Makefile 간단히 만들기

A terminal window titled 'hyejungko — cse20180305@cspro1: ~' displays the content of a Makefile. The text is as follows:

```
1 cc = gcc
2 target = main
3 object = dog.o blackcow.o turtle.o main.o
4
5 $(target): $(objects)
6     $(cc) -o $(target) $(objects)
7
8 $(objects) : animal.h
9
10 .PHONY: clean
11 clean:
12     rm $(target) $(objects)
```

2) gdb를 이용하여 코드의 문제를 찾고 이유를 설명하시오.

1) vi test.c

A terminal window titled 'hyejungko — cse20180305@cspro1: ~' displays the content of a C file named test.c. The text is as follows:

```
1 #include <stdio.h>
2
3 main(void)
4 {
5     int i;
6     double num;
7
8     for (i=0; i<5; i++) {
9         num=i/2 + i;
10        printf("num is %f \n", num);
11    }
12 }
```

```
[cse20180305@cspro1:~$ gdb test
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test...done.
```

```
[(gdb) l
1      #include <stdio.h>
2
3      main(void)
4      {
5          int i;
6          double num;
7
8          for (i=0; i<5; i++) {
9              num=i/2 + i;
10             printf("num is %f \n", num);
```

```
[(gdb) b 8
Breakpoint 1 at 0x40052e: file test.c, line 8.
[(gdb) r
Starting program: /sogang/under/cse20180305/test
```

for문에 이상이 있다고 판단하여 line 8에
breakpoint & run

```
Breakpoint 1, main () at test.c:8
8          for (i=0; i<5; i++) {
[(gdb) s
9              num=i/2 + i;
[(gdb) s
10             printf("num is %f \n", num);
```

step (s) 한 줄씩 실행

```
[(gdb) p num
$1 = 0
[(gdb) n
num is 0.000000
```

num 값 체크

```
8          for (i=0; i<5; i++) {
[(gdb) display i
1: i = 0
[(gdb) display num
2: num = 0
[(gdb) n
9              num=i/2 + i;
1: i = 1
2: num = 0
```

line 8 실행 후 i = 1 일 때 num 이 0이므로
잘못되었다.

num = (double) i/2 + i 로 변경해야 한다.

3) 프로그래밍 문제의 remove_blanks_at_the_end 함수를 작성하시오.

```
1 #include "Header.h"
2
3 void Remove_Blanks_At_The_End( char *line ) {
4     int i, k, newline_flag = 0;
5
6     // 전체 문장에 대하여 line[k] 가 줄바꿈 이면 flag를 1로 set, '\0'이면 break, 루프 탈출
7     for ( k = 0; ; k++ ) {
8         if ( line[k] == '\n' ) {
9             newline_flag = 1;
10            break;
11        }
12        else if (line[k] == '\0' ) {
13            break;
14        }
15    }
16
17    // 전체 문장에 대하여 line[i]가 공백문자가 아니라면 break, 루프 탈출
18    for ( i = k-1; i >= 0; i-- ) {
19        if (line[i] != ' ' ) {
20            break;
21        }
22    }
23
24    // flag가 1일 때 문장의 마지막은 줄바꿈 , 공백 , flag가 1이 아니면 '\0'
25    if ( newline_flag == 1 ) {
26        line[i+1] = '\n';
27        line[i+2] = '\0';
28    }
29    else {
30        line[i+1] = '\0';
31    }
32 }
```

2. remove_blank_and_the_end에 자세히 주석을 작성하시오. (코드 복붙 및 주석)

```
// 문자열 뒤의 연속된 공백을 제거하는 메소드
3 void Remove_Blanks_At_The_End( char *line ) {
4     // 변수 정의 및 0으로 초기화
        int i, k, newline_flag = 0;
5
6     // 전체 문장에 대하여 line[k] 가 줄바꿈이면 flag는 1, '\0'이면break, 루프 탈출
7     for ( k = 0; ; k++ ) {
8         if ( line[k] == '\n' ) {
9             newline_flag = 1;
10            break;
11        }
12        else if (line[k] == '\0' ) {
13            break;
14        } // k <- k+1
15    }
16
17 // 전체 문장에 대하여 line[i]가 공백문자가 아니라면 break, 루프 탈출
18    for ( i = k-1; i >= 0; i-- ) {
19        if (line[i] != ' ' ) {
20            break;
21        }
22    } // i <- i - 1
23
24 // newline_flag가 1일때 문장의 마지막은 줄바꿈, 공백
        newline_flag가 1이 아니면 '\0'
25    if ( newline_flag == 1 ) {
26        line[i+1] = '\n';
27        line[i+2] = '\0';
28    }
29    else {
30        line[i+1] = '\0';
31    }
32 } // 종료
```

3. 실습시간에 작성한 Makefile의 한줄 한줄의 의미를 설명하시오.

```
// gcc를 이용하여 컴파일
1 cc = gcc
// Makefile의 최종목표는 main
2 target = main
// make에서 사용되는 object file들의 모음
3 object = dog.o blackcow.o turtle.o main.o
4 // 명령어 정의 부분
5 $(target): $(objects)
6     $(cc) -o $(target) $(objects)
7 // 헤더 파일이 변경되어도 object 파일을 새로 생성한다.
8 $(objects) : animal.h
9
10 .PHONY: clean //예상치 못한 상황을 방지하기 위해 phony 사용
11 clean: // 현재 디렉토리에 있는 target 파일과 object 파일을 모두 지운다
12     rm $(target) $(objects)
```

4. make의 옵션들에 대해 정리하시오.

| | |
|---------|--|
| -C dir | Makefile을 읽지 말고 우선 dir로 이동한다. 순환 make에 사용된다 |
| -d | Makefile을 수행하면서 각종 정보를 출력한다. |
| -h | 옵션에 관한 도움말을 출력한다. (-help) |
| -f file | file에 해당하는 파일을 Makefile로 취급한다. |
| -r | 갖고 있는 규칙(suffix rule 등)을 없는 것으로 간주한다. 즉, 사용자가 규칙을 새로 정의한다. |
| -t | 파일의 생성 날짜를 현재 시간으로 갱신한다. |
| -v | make 버전을 출력한다. |
| -p | make에서 내부적으로 세팅되어 있는 값을 출력한다. |
| -k | 에러가 발생해도 멈추지 말고 계속 진행한다. |